THE OFFICE OF THE STATE CHIEF INFORMATION OFFICER
ENTERPRISE TECHNOLOGY STRATEGIES

North Carolina Statewide Technical Architecture

System Integration Domain

STATEWIDE TECHNICAL ARCHITECTURE

System Integration Domain

Initial Release Date:	September 11, 2003	Version:	1.0.0
Revision Approved Date:			
Date of Last Review:	March 17, 2004	Version	1.0.1
Date Retired:			
Architecture Interdependencies:			
Reviewer Notes: Reviewed and updated office title and copyright date. Added a hyperlink for the ETS email – March 17, 2004.			

Systems Integration - Integration Architecture

Principle 5.00.01 An Integration Architecture enables the inter-operation of multiple technologies into a single integrated network.

Rationale:

- Integration provides a bridge between the hetergeneous operational applications and platforms. An effective architecture ties together the mix of platforms, operating systems, transports, and applications.
- Integration of business applications between agencies and vendors or other agencies supports electronic commerce.

Systems Integration - Integration Architecture

Principle 5.00.02 An Integration Architecture addresses the correlating components of data interchange, business processing issues, and end-user presentation.

Rationale:

- The Integration Architecture encompasses the mulitple layers of new and existing systems and the middleware in between.

Systems Integration - Integration Architecture

Principle 5.00.03 An Integration Architecture meets the needs of linking heterogeous operational application systems while protecting existing investments.

Rationale:

- The Integration Architecture should take into account the need to use existing workstations, peripherals and existing transports to access existing and new applications.

Systems Integration - Integration Architecture

Principle 5.00.04 When making integration decisions, the life span of the solution is a key factor.

Rationale:

- Temporary solution may be engineered very differently than a long-term solution. Cost and effort need to be taken into consideration when providing a solution that is only needed on a temporary basis.
- Short-term solutions are often hard-wired and often have low performance. They are designed to be replaced or easily removed. Cost and effort should also be considered for a short-term solution.
- Long term solutions must be standardized, adaptable, and engineered for high performance.

Systems Integration - Integration Architecture

Principle 5.00.05 Integration Architecture relies on middle service tiers such as interface engines, database gateways, messaging, integration services, and third party tools.

Rationale:

- It is more cost effective and easier to maintain applications that use middle service tiers than to modify multiple legacy applications.
- New N-tier applications still need access to the legacy information stored throughout the enterprise.
- Refer to the Middleware chapter for more information middle service tiers not covered in this chapter.

Systems Integration - Integration Architecture

Principle 5.00.06 Minimize the impact to existing application systems.

Rationale:

- To the extent possible, the Integration Architecture should enable new applications to use existing resources with minimal disruption.
- Where possible, use non-invasive techniques for integration.
- Integration requires good communication infrastructure. If the basic network infrastructure is not in place, a single integrated network of application communication cannot be achieved. (Refer to the Network Domain.)

Systems Integration - Integration Architecture

Principle 5.00.07 Use statewide technologies whenever possible.

Rationale:

- To the extent possible, use the same technologies in the Integration Architecture that are used in the Statewide Technical Architecture.
- Limit the heterogeneity of the technology used in order to simplify integration and enable migration to future technologies.

Systems Integration - Integration Architecture

Principle 5.00.08 Provide maximum flexibility to integrate heterogeneous systems when enhancing existing end-user functionality through the use of a middle service tier.

Rationale:

- Implement the middle tier with standards whenever possible.

Systems Integration - Integration Architecture

Principle 5.00.09 Use existing integration solutions whenever possible.

Rationale:

- Instead of building a new integration technique from scratch, use an existing vendor solution that answers the specific integration needs of an application system.

Systems Integration - Integration Architecture

Principle 5.00.10 Include centralized security management as part of the Integration Architecture.

Rationale:

- Refer to the Security and Network Domain.

Systems Integration - Application Communication Middleware

Principle 5.00.11 Using application communication middleware is required in a heterogeneous, distributed environment.

Rationale:

- The tiers of a distributed application, which often run on different hardware and operating systems, must communicate.
- Application communication middleware enables both inter- and intra-application communications.

Systems Integration - Application Communication Middleware

Principle 5.00.12 Using message-oriented middleware changes the fundamental design for building distributed applications.

Rationale:

- Message allows asynchronous processing so applications can continue processing after a message is sent.

Systems Integration - Application Communication Middleware

Principle 5.00.13 Using remote procedure calls (RPCs) offer a good migration strategy.

Rationale:

- RPCs are the easiest transition for mainframe programmers. An RPC is simply a subroutine even though it is running a business rule on the network.
- RPCs are a mature technology. They are already bundled with many operating systems and databases.

Systems Integration - Application Communication Middleware

Principle 5.00.14 Minimize the use of distributed units of work.

Rationale:

- Distributed transaction monitors are becoming less and less a requirement as high speed networks and messaging subsystems are deployed.
- The need for a transaction monitor can be eliminated or reduced by using features of message oriented middleware, combined with application design.

Systems Integration - Application Communication Middleware

Principle 5.00.15 Do not use database middleware for application communication

Rationale:

- Database middleware has limited usefulness. It allows an application component to access data, thereby supporting a two-tier application design.
- Database middleware does not have the capability to provide all levels of inter-component communications. Stretching its use to inappropriate environments will ultimately result in systems that have performance problems.

Systems Integration - Application Communication Middleware

Principle 5.00.16 Using a broker facilitates reuse and shortens development cycles.

Rationale:

- A broker provides access to common services that can be reused and shared, thus reducing development costs.
- The state can reduce the resources spent on developing and maintaining "islands of applications," which include redundant code. Application developers can focus on new work rather than rework.
- New applications will be combined of new business rules and common shared business rules. Since part of the application is "pre-written" and "pre-tested," delivery of the total application should result more quickly.

Systems Integration - Application Communication Middleware

Principle 5.00.17 Precede selection of application development tools with an application communication middleware strategy.

Rationale:

- In the long term, use of middleware by many applications is of more strategic importance than any one application development tool. Middelware selection should drive the choice of application development tools, not vice versa.
- A range of communication methods is available through middleware. A combination of products may be required.

Systems Integration - Application Communication Middleware

Principle 5.00.18 Select third-party middleware rather than middleware supplied with a development tool.

Rationale:

- De-coupling the middleware from the application development tool provides more flexibility in changing development tools in the future. For example, integrated CASE tools often provide third-party message oriented middleware as well as their own, proprientary message oriented middleware.
- When given the choice of proprietary middleware versus third party middleware, select the third party middleware option. For instance, message oriented middleware provided by the

integrated CASE tool vendor limits flexibility and links to a specific vendor and product strategy more closely.

- If message oriented middleware is linked directly to a specific development package, then there is the risk of limited usefulness with other applications that are not developed with the same tool.

Systems Integration - Application Communication Middleware

Principle 5.00.19 Document application programming interfaces (APIs) and interface definition language (IDL).

Rationale:

- APIs and IDL for components and services must be documented so that developers know where they are and how to use them.
- For more information about components and services, refer to the Application and Componentware Architecture chapters.

Systems Integration - Application Communication Middleware Types

Best Practice 5.01.01 When possible, design applications to use asynchronous communication.

Rationale:

- Message oriented middleware supports asynchronous communications.
- Asynchronous messaging requires a distinctly different design. It is implemented with a very basic set of message oriented middleware commands.
- Message oriented middleware provides a reliable form of communication.
- Asynchronous communication offers more flexibility than synchronous communication. The downstream application has more control over its operation.

Systems Integration - Application Communication Middleware Types

Best Practice 5.01.02 Use Remote Procedure Calls (RPCs) when message oriented middleware is not available.

Rationale:

- RPCs provide an acceptable, albeit limited, method of communication between software components.
- RPCs require synchronous communication and are less efficient in the use of resources; they tie up resources from both the client and the server until the service has been provided.
- Synchronous communication requires error handling in the client application if the request is made while a server is unavailable.

Systems Integration - Application Communication Middleware Types

Best Practice 5.01.03 Use distributed transaction monitors only when distributed transactional integrity is required.

Rationale:

- Transaction processing monitors offer significant functionality in addition to transaction management. Using transaction processing monitors when other middleware services suffice may cause unnecessary overhead and may result in performance problems.
- Since distributed transactions are composed of multiple discreet functions, parts of transactions may be at risk for some period of time. Designing applications to eliminate the distributed units of work reduces the risk of transaction failure.
- When distributed transactional integrity is needed, use a TP monitor rather than the transaction management capability of a database management system.

Systems Integration - Application Communication Middleware Brokers

Best Practice 5.02.01 Manage a statewide broker as a strategic infrastructure component.

Rationale:

- The service broker is a critical part of the distributed computing environment because it allows the technical architecture to meet the three goals of efficiency, sharing of information and agency autonomy.
- Strategic infrastructure benefits all agencies and should be centrally managed.

Systems Integration - Application Communication Middleware Brokers

Standard 5.02.01 Use of the service broker is required for inter-application communication.

Rationale:

- The service broker was put in place due to the lack of standards for inter-application communication types such as RPC, MOM, and TP monitors. (See Technical Topic 1, Application Communication Middleware Types, the Standards section, for more information.)
- While the lack of standards is not an issue for development of any single application, it poses problems for communication between applications. The broker is proposed as a standard communication paradigm for inter-application communication.

Systems Integration - Application Communication Middleware Brokers

Best Practice 5.02.02 Be sure a statewide broker is independent of code development tools.

Rationale:

- The purpose of the service broker is to facilitate communication in a multi-platform, multi-language environment. If the service broker is tied to a single vendor's product, then the goal of facilitating communication in a diverse environment has not been met.
- Implementing a service broker that supports multiple vendors' products helps protect the state from being negatively impacted by market forces.

Systems Integration - Application Communication Middleware Brokers

Best Practice 5.02.03 Be sure a statewide broker provides a suite of communication middleware features.

Rationale:

- A best of breed approach should be taken when selecting the application communication middleware.

Systems Integration - Application Communication Middleware Brokers

Best Practice 5.02.04 Use the state's inter-application middleware, the service broker interface, for inter-application communication between state-developed applications. For interfaces with other applications, use the Interface Engine.

Rationale:

- State-developed applications gain performance and flexibility by using the service broker for inter-application communication.
- In-house or out-sourced custom-developed applications requiring inter-application communication should be capable of using a service broker. Applications sharing or requiring services from external application systems should provide the capability to use the standard inter-application communication middleware architecture.
- In instances where the application code cannot be modified, such as purchased applications where the state does not have rights to source code, use the interface engine. For more information about application integration, refer to the Integration Architecture chapter.
- For more information on the Interface Engine, see the Integration Architecture chapter.

Systems Integration - Application Integration

Standard 5.03.01 Clearly define Application Interfaces.

Rationale:

- To integrate applications for which the state has no source code rights, application interfaces must be clearly defined in order to allow reliable communication between applications.
- To facilitate puchase of best-of-breed software while easing application integration issues, the application interfaces must be clearly defined.

Systems Integration - Application Integration

Best Practice 5.03.01 Anticipate future usage.

Rationale:

- Whenever an application integration is constructed, anticipate future usage so the technology will be adaptable and scaleable.
- For example, an organization may not want to use a screen scraping interface if future requirements are for faster performance or additional information that is not supplied by an existing terminal screen.

Systems Integration - Application Integration

Standard 5.03.02 The message structure must be documented.

Rationale:

- A message or transaction is the mechanism for extracting data from an application or sending data to an application.
- Programmers integrating applications need to know record length and type (ie, whether it is a variable or fixed length record, and if it is variable, the delimiting characters used to separate the fields), and know which fields are optional versus required.
- A description of the data for each field is also necessary.
- Explanations and examples of record formats and field descriptions are helpful and should be included.

Systems Integration - Application Integration

Best Practice 5.03.02 Use application integration strategy for online transaction program (OLTP) application systems, not decision support systems (DSS).

Rationale:

- Data warehouses or other solutions shouuld be used in decision support applications. (For more information on data warehouses, refer to the Information Architecture chapter.)

Systems Integration - Application Integration

Standard 5.03.03 The application must be able to transmit and receive messages using a client/server model.

Rationale:

- The client is the process that sends or originates the message. The server is the process that receives the message.
- Clients and servers may communicate using TCP/IP and sockets, or other communication protocols, such as Serial and FTP, as long as they perform the same transmit and receive functionality.
- Packetization characters, which identify the start and end block strings, and message acknowledgement format must also be provided.

Systems Integration - Application Integration

Best Practice 5.03.03 Design an integration solution that does not write directly to an operational database.

Rationale:

- Existing application logic or busines rules should be used when updating an application database.
- An external user or application could inadvertently corrupt operational data.

Systems Integration - Application Integration

Best Practice 5.03.04 Consider a screen scraping solution when an application link needs to be non-invasive and there are no other non-invasive interfaces available to an application system.

Rationale:

- Using existing terminal screens can be a viable alternative to re-coding a legacy or purchased application for a program interface. Legacy and purchased application screen formats are normally static and contain the information needed by a new application.
- If screen scraping is used through an interface engine, it can be more reliable and stable than a PC screen scraping solution for operational applications.

Systems Integration - Application Integration

Standard 5.03.04 Purchase line-of-business application software rather than custom developing it whenever possible.

Rationale:

- Purchase line-of-business application software can permit the state to respond to business needs in a more timely manner than custom developing software.
- Published APIs are insuffient because their use requires custom development of state applications and it may be impossible to interface two purchased applications. Use of an interface engine provides maximum flexibility.

Systems Integration - Application Integration

Best Practice 5.03.05 Use direct program-to-program intefaces for high transaction volumes.

Rationale:

- Direct program-to-program interfaces pass only the required information between applications, so performance and throughput is at the optimal level.

Systems Integration - Application Integration

Best Practice 5.03.06 When designing an application integration solution using an interface engine, give careful consideration to the design and planning of the application interfaces and connectivity.

Rationale:

- At the beginning of the design stage, involve application developers who are knowledgeable in the business rules and interfaces to each system that needs to be accessed.
- Some application systems may have multiple entry or exit points that can be used. If a non-invasive solution is selected, capitalize on using the entry or exit points that best apply to your application needs.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.01 EDI should be fully integrated into the business process and the computer applications that support the process.

Rationale:

- The maximum benefits of EDI can be achieved when EDI is integrated into the business process.
- Focus on the business process supported by EDI rather than on the technology used by EDI.
- Maintain an audit trail that supports tracking and control for EDI transactions. Entries should be maintained for each handoff of a transaction between EDI application elements and between trading partners.
- Where practical, use reciprocal transactions to achieve application level acknowledgement of electronic transactions.

Systems Integration - Electronic Data Interchange (EDI)

Standard 5.04.01 Use ANSI X12 or UN-EDIFACT for Electronic Data Interchange (EDI).

Rationale:

In the United State, most ED transactions comply with ANSI standards approved by the ANSI X12 committee. These standards cover a wide range of commercial interaction, including, but not limited to the following:

- Requition, request for quotation, purchase order, purchase order change.
- Vehicle service order, product service claim, and product service claim response.
- Air freight information, motor carrier bill of lading, U.S. Customs status information, shipping instructions.
- Remittance, credit/debit adjustment, mortgage credit report, real estate title evidence, electronic filing to tax return.
- Student load application, student educational record, student enrollment verification.Most international EDI transactions comply with the UN-EDIFACT standards endorsed by the United Nations. EDIFACT standards are similar to ANSI X12 standards. EDIFACT standards cover EDI for administration, commerce, and transport. Note the ANSI s committed to migrating the X12 standards to be compatible with UN-EDIFACT standards.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.02 Use EDI to automate frequently used business transactions.

Rationale:

- EDI is cheaper, faster, and more accurate than performing the same transactions manually.
- EDI is appropriate for agency-to-agency transactions should exchange mapped data, without requiring the trading partners to go through the generation/interpretation process. This avoids the requirements that both internal trading partners buy and maintain EDI software.
- The state may require EDI.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.03 Use a single EDI software package for the entire enterprise, even if there are mulitple EDI servers.

Rationale:

- A single best-of-breed EDI software package reduces the complexity of implementing and managing EDI across the state.
- Business process and transaction volume may require additional installation of EDI software, but all should be the same EDI package.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.04 Use industry standard for transactions that are being performed electronically.

Rationale:

- For commerce between agencies, or between agencies and US-based trading partners, use the latest version of the ANSI X12 transaction set.
- Be aware that standards will evolve over time, and all participating trading partners will need to synchronize when a newer version is implemented.
- If there are no standards for transactions conducted frequently, define some with the major trading partners.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.05 Manage EDI as a critical application.

Rationale:

- EDI has a significant impact on the state's business. Even when EDI software is deployed on a PC-based server, it must be managed as a mission-critical application.
- Successful EDI implementations require transaction volume and capacity planning. (Refer to the System Management Architecture chapter for more information about capacity planning.)
- Secure EDI transactions with a level of security appropriate for the business function being performed.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.06 Use a value added network (VAN) for data transmission to outside trading partners.

Rationale:

- Direct file transfer (eg, "batch feeds") is acceptable for EDI performed within an agency or between agencies.
- Direct EDI introduces additional complexity into the transmission and receipt of transactions to outside trading partners.
- Look for opportuntity to use the Internet for EDI data transmission when standard software is available to handle Internet EDI.

- ❑ When performing financial EDI, also use an automated clearinghouse to manage the funds transfer portion of finanancial transactions.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.07 Purchase - do not build - software to perform EDI translation, formatting, and transmission to a VAN.

Rationale:

- ❑ Start small with a single transaction set to a single key trading partner; add additional trading partners and transactions after the others are running smoothly.

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.08 Use an EDI solution over an interface engine solution if possible.

Rationale:

- ❑ For EDI transaction sets, the EDI software has built-in capability to format transaction data.
- ❑ When using EDI, only one interface program is need: the originating application (for outgoing transactions) or the target application (for incoming transactions).

Systems Integration - Electronic Data Interchange (EDI)

Best Practice 5.04.09 If EDI software does not include data mapping capability, use an interface engine, rather than custom programming for data mapping.

Rationale:

- ❑ Data mapping using an interface engine is noninvasive.

Systems Integration - Data Access Integration

Best Practice 5.05.01 Use as few middleware layers as possible when implementing a database gateway.

Rationale:

- ❑ Additional layers of middleware in between an application and the database gateway could hinder performance of mission critical applications. For example, an application that needs to access a database gateway can implement an ODBC middleware layer that ultimately accesses the gateway middleware. Application performance can be increased if the application was written to make direct calls to the gateway middleware, omitting the ODBC layer.
- ❑ If there are fewer middle conversion tiers, there are less operational layers to maintain in the event of maintenance or upgrades. For example, if there is a change to an application database location, or an upgrade or maintenance update to the middleware software, it can effect all end user workstations and servers that access that application.

Systems Integration - Data Access Integration

Standard 5.05.01 There is no Remote Procedure Call (RPC) standard. Use the state of North Carolina's service broker for inter-application communication.

Rationale:

- Even with an RPC that is endorsed by a vendor neutral party, such as the Open Group, there is no standard RPC.
- RPCs are available from different vendors, such as the Open Group's DCE RPC, Sun Microsystem's ONC/RPC, and Microsoft's RPC.
- Each vendors version has a different application programming interface and they do not inter-operate with one another.

Systems Integration - Data Access Integration

Best Practice 5.05.02 Balance the type of data access method implemented with required performance needed by the application end users and the impact to the existing operational databases.

Rationale:

- If the wrong data access method is selected, the performance may not match the application needs.
- A solution that is good for a new application may adversely impact existing operational applications.

Systems Integration - Data Access Integration

Standard 5.05.02 There is no Message Oriented Middleware (MOM) standard. Use the state of North Carolina's service broker for inter-application communication.

Rationale:

- At present, all message oriented middleware is proprietary. Products from different vendors have different application programming interfaces, which do not inter-operate with one another.

Systems Integration - Data Access Integration

Best Practice 5.05.03 Keep the integration strategy as simple as possible.

Rationale:

- The more complicated the strategy, the more difficult it is to maintain and change.

Systems Integration - Data Access Integration

Standard 5.05.03 There is no distributed transaction processing (TP) monitor standard. Use the state of North Carolina's service broker for inter-application communication.

Rationale:

- The applications coordinated by a tranacation monitor which will run on different platforms with access to different databases and resource managers.
- The applications are often developed using different tools and have no knowledge of one another.
- Industry standards specify how a TP monitor interfaces to resource managers, other TP monitors, and its clients.
- X/Open XA specification defines specfications for two-phase commits that work with distributed databases.
- X/Open TX standard defines transactions.
- X/Open X/ATMI provides a standard transaction management interface.

Systems Integration - Data Access Integration

Best Practice 5.05.04 Code data integrity verification rules into the DBMS whenever possible, particularly when external users and programs will be writing data directly to the DBMS.

Rationale:

- Since most DBMS vendors can code triggers and rules into the database, it is recommended to use this technology wherever possible in order to ensure data integrity.
- For more information on databases, refer to the Data Architecture chapter.

Systems Integration - Data Access Integration

Best Practice 5.05.05 Separate decision support systems (DSS) from online transaction processing (OLTP) database whenever possible.

Rationale:

- If this practice is feasible, it will reduce the impact of ad hoc and large queries from decision support systems onto production operational application databases that are used by online users for day-to-day operations.
- For more information on database architecture, refere to the Data Architecture chapter.

Systems Integration - Terminal Integration

Best Practice 5.06.01 Design new application systems with the user interface as a separate application tier.

Rationale:

- If a GUI is not designed as a separate tier, and a character-based terminal interface to the client/server application is not possible, terminal integration will not be successful.
- When building a client/server application that needs to be accessed by a wide variety of end users and platforms, ensure that it can be accessed by any type of user interface, including graphical user interface and character-base interfaces.

Systems Integration - Terminal Integration

Best Practice 5.06.02 Base the user interface design on the targeted end user platform.

Rationale:

- If the inteface is similar to existing user interfaces, the learning curve is reduced and the end users can easily become productive on new systems.
- If a GUI-based interface is used, conform to established GUI-based standards.
- If a 3270 terminal interface is used, design the interface similar to existing screen interfaces that are used by legacy systems.

Systems Integration - Terminal Integration

Best Practice 5.06.03 Implement as few comminications tiers are possible.

Rationale:

- With fewer tiers the architecture will be simpler and easier to maintain.